Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L17: Entry 1 of 12 File: USPT Nov 14, 2000

DOCUMENT-IDENTIFIER: US 6147767 A
TITLE: Computer system for a printer which stores and operates multiple application programs

Application Filing Date (1):
19970905

Detailed Description Text (8):
Referring to FIG. 3, the main operating system program, Program/File monitor, is shown. CPU 12 runs Program/File Monitor when power from source 16 is first supplied to controller 10. This may occur when source 16 is connected in the printer, or by an on/off switch which enables power to be supplied from source 16 to CPU 12. Upon such power-up, CPU 12 initializes its various components (step 46). CPU 12 first initializes its I/O ports, and checks the validity of memory 28 block by block. To check memory validity, if a checksum value embedded in each block of memory 28 matches the actual checksum of bytes in each block, the block is considered valid. CPU 12 generates a list of all valid files it finds as it checks the memory 28 validity based on the files unique file names. CPU 12 executes files 38, 39 and 41 to set up the communication driver to interface 18, 19 or 20, display 30 and keypad 22, and printer driver, respectively. CPU 12 also sets a RESET flag in memory 28. This reset flag may also be set if CPU 12 detects a pre-determined key sequence entered via keypad 22 representing a reset command from the user. If a RESET flag is set, CPU 12 branches to step 50 and searches the list generated at step 46 for valid application program files, i.e., files in which all their blocks of memory 28 were determined valid at step 46.

Detailed Description Text (15):
Referring to FIG. 5, the operation of step 59 (FIG. 4) for loading a new application program file will be described. Responsive to receiving a load new program file command, CPU 12 requests from the host computer the file size (step 62). CPU 12 then checks if the file to be loaded replaces an existing file in memory (step 64). For example, this may be done by comparing the name of file to be loaded, which may be received in the load new program/file command, with the names of application program file in the list generated at step 50 (FIG. 4). If a match occurs, then the file is a replacement file, and CPU 12 branches to step 65. At step 65, CPU 12 checks if the size of the file obtained at step 62 is greater than available space in temporary storage of memory 28. Memory 28 includes space allocated for temporary storage of new application programs. If not, CPU 12 instructs the host computer to send the new file, and writes the received file into temporary storage (step 68). CPU 12 then checks the integrity of the loaded file by performing a checksum operation on each of its blocks (step 70), similar to that described at step 46 (FIG. 4). If the file checks OK (step 72), CPU 12 deletes the existing file of the same filename from memory 28, and the new file is loaded from temporary storage in its place in memory 28 (step 74). This optimizes storage space for application programs in memory 28 by avoiding duplicative files. From step 74, CPU 12 proceeds to step 76, and the loading of the new file is complete. If the checksum operation fails at step 70, CPU 12 branches from step 72 to step 76, and the new file is not stored in memory 28.

Detailed Description Text (22):
The scan data capture event subroutine translates the consecutive stored running timer values into barcode data. At step 100 of FIG. 7C, CPU 12 calculates the width of the bars and white spaces between bars from the timer values captured by the DMA in memory. Widths are calculated by subtracting the value of each two consecutive captured timer values from memory. These calculated widths correspond to the actual widths of the bars and white spaces between bars of the barcode, because the transitions in scan data signal occur proportionally with the time the scanning beam transverses the barcode. The calculated widths represent captured data. This captured data is run through an enabled (executed) barcode decode routine for translating the calculated widths into decoded barcode data (step 102). The decode routine may be typical of routines used to decode widths of bars and white spaces into barcode data based upon the particular encoding format of the bar code. If the barcode is not successfully decoded at step 102, the subroutine of FIG. 7C exits to step 96 (FIG. 6) since decode was incomplete (FIG. 6). If needed, checksum verification and redundancy decode checks are performed on decoded barcode data from step 102 to assure that the decoded barcode data is valid (step 106, 108 and 110). Failure of these checks means that decode was incomplete, causing the subroutine of FIG. 7C to exit to step 96. Upon a successful decode of the captured data into barcode data, the decoded barcode data is transferred to an output queue for subsequent storage in memory 28. If in 2-D scan mode, the captured data represents the data of the row scanned, which is then stored as row/column data in memory 28.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L17: Entry 2 of 12 File: USPT Jun 29, 1999

DOCUMENT-IDENTIFIER: US 5918047 A
** See image for Certificate of Correction **
TITLE: Initializing a processing system

Application Filing Date (1):
19970122

Detailed Description Text (19):
At step 112 an image of flash memory 20 is copied performing a checksum. At step 114, method 100 determines if the checksum result is correct. This step tests the overall validity of the primary code blocks contained in flash memory 20. If the checksum result is not correct, then at step 116, method 100 reads secondary storage 22. At step 118, flash memory 20 is reloaded with the alternate code blocks--alternate boot block code 28 and alternate BIOS codes blocks 30--from secondary storage 22. Method 100 then returns to step 112 where the image of flash memory 20 is copied performing a checksum.

Detailed Description Text (20):
Alternatively, if the checksum result is correct, method 100 proceeds to step 120 where primary boot block code 24 is copied into shadow RAM 18. The primary boot block code includes a table. At step 124, the table is read. The table describes how shadow RAM 18 should be constructed. In other words, the table lists the addresses for the BIOS code blocks 26 that should be copied into shadow RAM 18 when processing system 10 is initialized. As explained below, the table enables method 100 to test the validity of each primary BIOS code block 26 before that code block is copied into shadow RAM 18.

Detailed Description Text (23):
At step 134, method 100 determines if the primary BIOS code block is defective. If the primary BIOS code block is not defective, method 100 proceeds to step 136 where it is determined whether the primary BIOS code block should be loaded. If the primary BIOS code block should be loaded, method 100 proceeds to step 140 where the BIOS code block is copied into shadow RAM 20 performing a checksum. If the primary BIOS code block should not be loaded, method 100 returns to step 124 where the table is read.

Detailed Description Text (24):
Alternatively, if it is determined at step 134 that the primary BIOS code block is defective, method 100 determines whether the corresponding alternate BIOS code block should be loaded at step 138. If the alternate BIOS code block should be loaded, method 100 replaces the primary BIOS code block in flash memory 20 with the corresponding alternate BIOS code block from secondary storage 22 at step 139. Method 100 then proceeds to step 140 where the primary BIOS code block is copied into shadow RAM 18 performing a checksum. Otherwise, if the alternate BIOS code block should not be loaded, method 100 returns to step 124 where the table is read.

Detailed Description Text (25):
At step 142, method 100 determines if the checksum result is correct. If the

checksum is not correct, method 100 returns to step 116 where the secondary storage 22 is read. On the other hand, if checksum is correct, method 100 returns to step 124 where the table is read.

CLAIMS:

4. The method of claim 1, wherein the step of testing comprises the step of verifying a checksum of the primary code block.

12. The method of claim 9, wherein the step of testing the primary BIOS code block comprises the step of examining a checksum of the primary BIOS code block.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L17: Entry 3 of 12 File: USPT Dec 31, 1996

DOCUMENT-IDENTIFIER: US 5590329 A
TITLE: Method and apparatus for detecting memory access errors

Application Filing Date (1):
19940204

Detailed Description Text (121):
Due to the slow operating speeds normally associated with memory allocation functions, such as malloc, many programmers will often allocate one large block of memory, and then break the large block into smaller pieces, as needed, with each smaller piece of allocated memory being accessible by at least one pointer. However, according to the pointer table updating processes outlined above, the valid range for each pointer would normally be recorded in the pointer check table 200 as the entire large block of memory.

Detailed Description Text (197):
The evaluator 35 will then perform a cyclic redundancy check (CRC) checksum during step 1220 on all of the uninitialized bytes that may be accessed by the compiled code. During step 1225 the compiled function is called with the proper arguments and executed. Following execution of the compiled code, program control will return to step 1230, wherein a second CRC checksum is performed on all of the bytes that were accessible by the compiled code and uninitialized before execution of the compiled code.

Detailed Description Text (198):
A test is performed during step 1235 to determine if the value of any byte has changed by testing if the results of the CRC checksum that was performed during step 1220 equals the results of the CRC checksum that was performed during step 230. If it is determined during step 1235, that the two checksum values are unequal, then program control will proceed to step 1240.

Previous Doc Next Doc Go to Doc#

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Thursday, October 14, 2004

Hide?	Set Name	Query	Hit Count
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
☐	L17	L14 and checksum	12
☐	L16	L14 and (address near8 outside near8 range)	0
☐	L15	L14 and (((return or returned) near4 address) near8 outside near8 range)	0
☐	L14	20000516	73
☐	L13	(check or checking or test or testing) near8 (valid or validity) near8 (memory near5 (region or partition or block))	90
☐	L12	L8 and (((return or returned) near4 address) near8 outside near8 range)	0
☐	L11	L10 and l9	0
☐	L10	L8 and (((return or returned) near4 address) near8 dedicated)	4
☐	L9	L8 and checksum	181
☐	L8	20000516	3750
☐	L7	L6 and checksum	294
☐	L6	(check or checking or valid or validity or test or testing)near8 (memory near5 (region or partition or block))	5814
☐	L5	(check or checking or valid or validity or test or testing) same (memory near5 (region or partition or block))	14605
☐	L4	(check or checking or valid or validity or test or testing) same chechsum same (memory near5 (region or partition or block))	0
	DB=USPT; PLUR=YES; OP=ADJ		
☐	L3	L1 and (check or checking or valid or validity or test or testing)	0
☐	L2	L1 and (check or checking or valid or validity)	0
☐	L1	4109311.pn.	1

END OF SEARCH HISTORY